PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 15/80

A1

(11) International Publication Number: WO 92/00572

(43) International Publication Date: 9 January 1992 (09.01.92)

(21) International Application Number: PCT/GB91/01053

(22) International Filing Date: 28 June 1991 (28.06.91)

(30) Priority data:
9014569.9 29 June 1990 (29.06.90) GB

(71) Applicants *(for all designated States except US)*: UNIVERSITY COLLEGE LONDON [GB/GB]; Gower Street, London WC1E 6BT (GB). KING'S COLLEGE LONDON [GB/GB]; Strand, London WC2R 2LS (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)* : TAYLOR, John, G. [GB/GB]; 33 Meredyth Road, Barnes, London SW13 0DS (GB). GORSE, Denise [GB/GB]; 15C Argyle Road, Ilford, Essex IG1 3DH (GB). CLARKSON, Trevor, Grant [GB/GB]; 83 College Park Close, London SE13 5EZ (GB).

(74) Agent: BOON, Graham, Anthony; Elkington and Fife, Prospect House, 8 Pembroke Road, Sevenoaks, Kent TN13 1XR (GB).

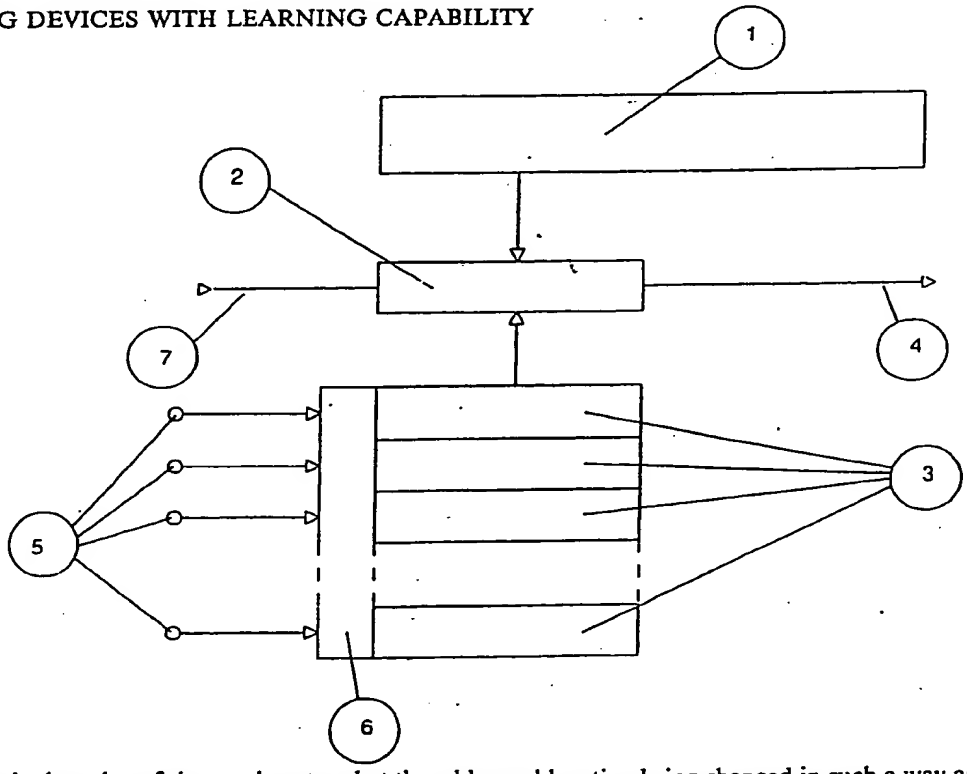
(81) Designated States: AT, AT (European patent), AU, BB, BE (European patent), BF (OAPI patent), BG, BJ (OAPI patent), BR, CA, CF (OAPI patent), CG (OAPI patent), CH, CH (European patent), CI (OAPI patent), CM (OAPI patent), CS, DE, DE (European patent), DK, DK (European patent), ES, ES (European patent), FI, FR (European patent), GA (OAPI patent), GB, GB (European patent), GN (OAPI patent), GR (European patent), HU, IT (European patent), JP, KP, KR, LK, LU, LU (European patent), MC, MG, ML (OAPI patent), MN, MR (OAPI patent), MW, NL, NL (European patent), NO, PL, RO, SD, SE, SE (European patent), SN (OAPI patent), SU, TD (OAPI patent), TG (OAPI patent), US.

Published
With international search report.

(54) Title: NEURAL PROCESSING DEVICES WITH LEARNING CAPABILITY

(57) Abstract

A neuron for use in a neural processing network, comprises a memory having a plurality of storage locations (3) at each of which a number representing a probability is stored, each of the storage locations being selectively addressable to cause the contents of the location to be read to an input of a comparator (2). A noise generator (1) inputs to the comparator a random number representing noise. At an output of the comparator an output signal (4) appears having a first or second value depending on the values of the numbers received from the addressed storage location and the noise generator, the probability of the output signal having a given one of the first and second values being determined by the number at the addressed location. The neuron receives from the environment signals (r, p) representing success or failure of the network, the value of the number stored at the addressed location being changed in such a way as to increase the probability of the successful action if a success signal is received, and to decrease the probability of the unsuccessful action if a failure signal is received.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MG	Madagascar
AU	Australia	FI	Finland	ML	Mali
BB	Barbados	FR	France	MN	Mongolia
BE	Belgium	GA	Gabon	MR	Mauritania
BF	Burkina Faso	GB	United Kingdom	MW	Malawi
BG	Bulgaria	GN	Guinea	NL	Netherlands
BJ	Benin	GR	Greece	NO	Norway
BR	Brazil	HU	Hungary	PL	Poland
CA	Canada	IT	Italy	RO	Romania
CF	Central African Republic	JP	Japan	SD	Sudan
CG	Congo	KP	Democratic People's Republic of Korea	SE	Sweden
CH	Switzerland			SN	Senegal
CI	Côte d'Ivoire	KR	Republic of Korea	SU	Soviet Union
CM	Cameroon	LI	Liechtenstein	TD	Chad
CS	Czechoslovakia	LK	Sri Lanka	TG	Togo
DE	Germany	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		

NEURAL PROCESSING DEVICES WITH LEARNING CAPABILITY

This invention relates to artificial neuron-like devices (hereinafter referred to simply as "neurons") for use in neural processing.

One of the known ways of realising a neuron in practice is to use a random access memory (RAM). The use of RAMs for this purpose dates back a considerable number of years. Recently, a particular form of RAM has been described (see Proceedings of the First IEE International Conference on Artificial Neural Networks, IEE, 1989, No. 313, pp 242-246) which appears to have the potential for constructing neural networks which mimic more closely than hitherto the behaviour of physiological networks. This form of RAM is referred to as a pRAM (probabilistic random access memory). For a detailed discussion of the pRAM attention is directed to the paper identified above. However, a brief discussion of the pRAM is set out below, by way of introduction to the invention.

The pRAM is a hardware device with intrinsically neuron-like behaviour (Figure 1). It maps binary inputs [5] (representing the presence or absence of a

pulse on each of N input lines) to a binary output [4] (a 1 being equivalent to a firing event, a 0 to inactivity). This mapping from $(0,1)^N$ to (0,1) is in general a stochastic function. If the 2^N address locations [3] in an N-input pRAM A are indexed by an N-bit binary address vector $\underline{u}$, using an address decoder [6], the output $a \in (0,1)$ of A is 1 with probability

$$\text{Prob}(a=1 \mid \underline{i}) = \sum_{\underline{u}} \alpha_{\underline{u}} \prod_{j=1}^{N} (i_j u_j + \bar{i}_j \bar{u}_j) \qquad (1)$$

where $\underline{i} \in (0,1)^N$ is the vector representing input activity (and $\bar{x}$ is defined to be 1-x for any x). The quantity $\alpha_{\underline{u}}$ presents a probability. In the hardware realisation of the device $\alpha_{\underline{u}}$ is represented as an M-bit integer in the memory locations [3], having a value in the range 0 to 2^M-1 and these values represent probabilities in the range

$$(0, \frac{1}{2^M}, \frac{2}{2^M}, \ldots, 1-\frac{1}{2^M}).$$

The $\alpha_{\underline{u}}$ may be assigned values which have a neuro-biological interpretation: it is this feature which allows networks of pRAMs, with suitably chosen memory contents, to closely mimic the behaviour of living neural systems. In a pRAM, all 2^N memory components are independent random variables. Thus, in addition to possessing a maximal degree of non-linearity in its response function - a deterministic ($\underline{\alpha} \in (0,1)^N$) pRAM can realise any of the

2^{2^N} possible binary functions of its inputs - pRAMs differ from units more conventionally used in neural network applications in that noise is introduced at the synaptic rather than the threshold level; it is well known that synaptic noise is the dominant source of stochastic behaviour in biological neurons. This noise, ν, is introduced by the noise generator [1]. ν is an M-bit integer which varies over time and is generated by a random number generator. The comparator [2] compares the value stored at the memory location being addressed and ν. One way of doing this is to add the value stored at the addressed location to ν. If there is a carry bit in the sum, i.e. the sum has M+1 bits, a spike representing a 1 is generated on arrival of the clock pulse [7]. If there is no carry bit no such spike is generated and this represents a 0. It can be seen that the probability of a 1 being generated is equal to the probability represented by the number stored at the addressed location, and it is for this reason that the latter is referred to as a probability. It should be noted that the same result could be achieved in other ways, for example by generating a 1 if the value of the probability was greater than ν. It can also be noted that because pRAM networks operate in terms of 'spike trains' (streams of binary digits produced by the addressing of successive memory locations) information about the timing of firing

events is retained; this potentially allows phenomena such as the observed phase-locking of visual neurons to be reproduced by pRAM nets, with the possibility of using such nets as part of an effective 'vision machine'.

For information concerning in particular the mathematics of the pRAM attention is directed to the paper written by the present inventors in the Proceedings of the First IEE International Conference in Artificial Neural Networks, IEE, 1989, No. 313, pp. 242-246, the contents of which are incorporated herein by reference.

Figure 9 shows a simple neural network comprising two pRAMs denoted as RAM 1 and RAM 2. It will be understood that for practical applications much more extensive networks are required, the nature of which depends on the application concerned. Nevertheless, the network shown in Figure 9 illustrates the basic principles. It will be seen that each pRAM has an output OUT and a pair of inputs denoted IN1 and IN2. Each output corresponds to the output [4] shown in Figure 1. The output from RAM 1 is applied as an input IN1 of RAM 1, and the output from RAM 2 is applied as an input to the input IN2 of RAM 1. The output from RAM 1 is also applied as an input to the input IN2 of

RAM 2, and the output of RAM 2 is applied as an output to the input IN1 of RAM 2. The network operates in response to clock signals received from the circuit labelled TIMING & CONTROL.

The circuitry of RAM 1 is shown in detail in Figure 10. RAM 2 is identical, except that for each reference in Figure 10 to RAM 1 there should be substituted a reference to RAM 2 and vice versa.

RAM 1 comprises a random number generator. This is of conventional construction and will therefore not be described here in detail. The embodiment shown here employs shift registers and 127 stages are used to give a sequence length of $2^{127}-1$. It will be noted that the random number generator has an array of three EXOR gates having inputs 2, 3 and 4 which can be connected to selected ones of the taps T of the shift registers. The taps selected in RAM 1 will be different to those selected in RAM 2 and appropriate selection, according to criteria well known to those in the art, avoids undesired correlation between the random numbers generated by the two generators. The output of the random number generator is an 8-bit random number which is fed as two 4-bit segments to two adders which make up a comparator.

The illustrated embodiment has a memory which holds four 8-bit numbers held at four addresses. The memory is thus addressed by 2-bit addresses. At each operation of the network the contents of the addressed storage location in the memory are fed to the comparator where they are added to the random number generated at that time. The output of the comparator is a '1' is the addition results in a carry bit and is a '0' otherwise.

The output of the comparator is fed to the output of the RAM (which is labelled OUT in Figure 9) and also to a latch. Here it is held ready to form one bit of the next address to be supplied to the address decoder via which the memory is addressed. As can be seen by taking Figures 9 and 10 together, the other bit of the address (i.e. that supplied to input IN2 of RAM 1) is the output of RAM 2.

Figure 10 also shows inputs labelled R1_LOAD and MEMORY DATA which enable the system to be initialised by loading data into the memory at the outset, and an input SCLK by means of which clock pulses are supplied to RAM 1 from a clock generator (see below). Finally as regards Figure 10, there is an input denoted GENERATE which is connected to the latch via an inverter gate which serves to initiate the production

of a new output from the pRAM and allows a set of 8 SCLK pulses to occur. The clock generator shown in Figure 11 is of conventional construction and will therefore not be described in detail, its construction and operation being self-evident to a man skilled in the art from the Figure. This provides a burst of 8 clock signals at its output SCLK which is supplied to the timing input of each of RAM 1 and RAM 2. Each time a GENERATE pulse occurs, each of RAM 1 and RAM 2 generates a new 8-bit random number (one bit for each SCLK pulse), addresses a given one of the four storage locations in its memory, compares the random number with the contents of the addressed location with the random number, and generates an output accordingly.

Reinforcement training is a strategy used in problems of adaptive control in which individual behavioural units (here to be identified with pRAMs) only receive information about the quality of the performance of the system as a whole, and have to discover for themselves how to change their behaviour so as to improve this. Because it relies only on a global success/failure signal, reinforcement training is likely to be the method of choice for 'on-line' neural network applications.

A form of reinforcement training for pRAMs has been

devised which is fast and efficient (and which is capable, in an embodiment thereof, of being realised entirely with pRAM technology). This training algorithm may be implemented using digital or analogue hardware thus making possible the manufacture of self-contained 'learning pRAMs'. Networks of such units are likely to find wide application, for example in the control of autonomous robots. Control need not be centralised; small nets of learning pRAMs could for example be located in the individual joints of a robot limb. Such a control arrangement would in many ways be akin to the semi-autonomous neural ganglia found in insects.

According to the present invention in its preferred form there is therefore provided a device for use in a neural processing network, comprising a memory having a plurality of storage locations at each of which a number representing a probability is stored; means for selectively addressing each of the storage locations to cause the contents of the location to be read to an input of a comparator; a noise generator for inputting to the comparator a random number representing noise; means for causing to appear at an output of the comparator an output signal having a first or second value depending on the values of the numbers received from the addressed storage location and the noise

generator, the probability of the output signal having a given one of the first and second values being determined by the number at the addressed location; means for receiving from the environment signals representing success or failure of the network; means for changing the value of the number stored at the addressed location if a success signal is received in such a way as to increase the probability of the successful action; and means for changing the value of the number stored at the addressed location if a failure signal is received in such a way as to decrease the probability of the unsuccessful action. The number stored at the addressed location may be changed by an appropriate increment or decrement operation, for example.

A preferred form of the training rule represented by this aspect of the invention is described by the equation

$$\Delta\alpha_{\underline{u}}(t) = \rho((a - \alpha_{\underline{u}})r + \lambda(\bar{a} - \alpha_{\underline{u}})p\ )(t).\delta\ (\underline{u} - \underline{i}(t)) \quad (2)$$

where r(t), p(t) are global success, failure signals $\in$ {0,1} received from the environment at time t, the environmental response might itself be produced by a pRAM, though it might be produced by many other things). a(t) is the unit's binary output, and ρ, λ are constants $\in$[0,1]. The delta function is included

to make it clear that only the location which is actually addressed at time t is available to be modified, the contents of the other locations being unconnected with the behaviour that led to reward or punishment at time t. When r = 1 (success) the probability $\alpha_{\underline{u}}$ changes so as to increase the chance of emitting the same value from that location in the future, whilst if p = 1 (failure) the probability of emitting the other value when addressed increases. The constant λ represents the ratio of punishment to reward; a non-zero value for λ ensures that training converges to an appropriate set of memory contents and that the system does not get trapped in false minima. Note that reward and penalty take effect independently; this allows the possibility of 'neutral' actions which are neither punished or rewarded, but may correspond to a useful exploration of the environment.

In the accompanying drawings:

Figure 1 shows diagrammatically a pRAM, as described above;

Figure 2 shows diagrammatically an embodiment of a pRAM having learning characteristics according to the present invention;

Figure 3 shows an alternative embodiment of a pRAM having learning characteristics;

Figure 4 shows diagrammatically a pRAM adapted to handle a real-valued input;

Figure 5 shows diagrammatically a pRAM having the ability to implement a more generalised learning rule than that employed in Figure 2;

Figure 6 shows diagrammatically a pRAM in which eligibility traces (explained below) are added to each memory location;

Figure 7 shows how a pRAM with eligibility traces can be used to implement Equation 9(a) (for which see below);

Figure 8 shows the further modifications needed to implement Equation 10 (for which see below);

Figure 9 shows a simple neural network using two pRAMs;

Figure 10 is a circuit diagram showing one of the pRAMs of Figure 9 in detail; and

Figure 11 is a circuit diagram showing the timing and

control circuitry used in Figure 9.

Figure 2 shows one way in which rule (2) can be implemented in hardware. The memory contents $\alpha_{\underline{i}}(t+1)$ are updated each clock period according to rule (2). The pRAM [8] is identical to the unit shown in Figure 1 and described in the text above. For a given address on the address inputs [5], an output spike is generated as described above. The terms $a - \alpha_{\underline{u}}$ and $\overline{a} - \alpha_{\underline{u}}$ are produced using the inverter [11] and the adder/subtractors [12] where $\alpha_{\underline{u}}$ is read from the pRAM memory port [9]. These terms are multiplied by the reward and penalty factors ρr [14] and $\rho\lambda p$ [15] respectively using the multipliers [13]. The resultant reward/penalty increment is added to the value stored at the location being addressed [9] using a further adder [12] and is then written back into the memory using the write port [10].

The learning rule (2) achieves a close approximation to the theoretically expected final values of the memory contents for a suitably small value of the learning rate constant ρ. However, this may lead to a lengthy time for training. To increase the training speed, ρ may be initially set to a large value and subsequently decremented at each successive time step by a factor which vanishes suitably fast as the number of steps

increases.

The rule (2) may also be realised in hardware using a pRAM technology (Figure 3). The advantages of this method is that multiplier circuits are not required. However, this requires 2^M cycles to generate $\alpha_{\underline{i}}(t+1)$ where M is the number of bits used to represent $\alpha_{\underline{u}}$. It is implementable, in this example, by an auxiliary 4-input pRAM [16] (Figure 3) with input lines carrying $\alpha_{\underline{i}}(t)$, a(t), r(t) and p(t), (the order of significance of the bits carried by lines going from α_i to p) and with memory contents given by

$$\beta = (0,0,0,0,\rho\lambda,0,0,1-\rho\lambda,0,1-\rho,\rho,1,\rho\lambda,1-\rho,\rho,1-\rho\lambda) \quad (3)$$

Because $\alpha_{\underline{i}}(t) \in [0,1]$, and pRAMs are neuron-like objects which communicate via discrete pulses, it is necessary to use time-averaging (over a number of cycles, here denoted by R) to implement the update. The output [17] of the auxiliary pRAM [16] in each step consists of the contents of one of two locations in pRAM [16], since a, r and p remain the same and only α_i alters between 0 and 1. The output of the pRAM [16] accumulated over R time steps using the integrator [19] is the updated memory content $\alpha_{\underline{i}}(t+1) = \alpha_{\underline{i}}(t) + \Delta\alpha_{\underline{i}}(t)$, where $\Delta\alpha_{\underline{i}}(t)$ is given by (2). The memory location is updated with the integrator output using the write memory port [10]. It is simplest to set $R = 2^M$, where M is the number of bits used to represent the $\alpha_{\underline{u}}$'s. The steps used in the

update are

0. Set contents of M-bit register [19] to zero.

1. Record $\underline{i}$ (t) (location addressed), a(t) (using the latch [18], and r(t) and p(t) (the reward [24] and penalty [25] signals). [20] represents the 'environment' which provides the reward and penalty signals.

2. For the next R time step repeatedly address the same location $\underline{i}$ in pRAM [8] (to produce the spike train $\alpha_{\underline{i}}$). Let these pulses, together with the recorded a, r and p, generate spikes from locations in the auxiliary pRAM [16] and accumulate these values in the integrator [19].

3. [19] now contains an M-bit approximation to $\alpha_{\underline{i}}(t+1)$. Copy this into location $\underline{i}$ of pRAM [8] using port [10].

When the pRAM is implemented using analogue circuitry, [19] becomes an integrator which is first cleared and then integrates over R time steps. The output after this period is then written into the pRAM address $\underline{i}$. This is functionally identical to the description of the digital device above.

The ability to let the learning rate, ρ, decrease with time, as described in association with Figure 2, may also be included in the method of Figure 3.

There are many interesting problems of adaptive control which require real-valued inputs. An object of a further aspect of the invention is to provide a modified pRAM which enables such inputs to be handled.

Our copending application filed on even date herewith under the title "Neural processing devices for handling real-valued inputs" describes such pRAMs.

According to the invention of the copending application there is provided a neuron for use in a neural processing network, comprising a memory having a plurality of storage locations at each of which a number representing a probability is stored; a real number-to-digital converter which receives a plurality of real-valued numbers each in the range 0 to 1 and produces at its output a corresponding plurality of synchronised parallel pulse trains which are applied to the respective lines of the memory to define a succession of storage location addresses, the probability of a pulse representing a 1 being present in an address on a given address line being equal to the value of the real-valued number from which the pulse train applied to that address line was derived; a comparator connected to receive as an input the contents of each of the successively addressed locations, a noise generator for inputting to the

comparator a succession of random numbers representing noise; means for causing to appear at an output of the comparator a succession of output signals each having a first or second value depending on the values of the numbers received from the addressed storage locations and the noise generator, the probability of a given output signal having a given one of the first and second values being determined by the number at the addressed location; and an integrator for integrating the output signals from the comparator.

The device provided by the invention of the copending application performs mappings from $[0,1]^N$ to (0,1) using ideas of time-averaging similar to those used above to implement the reinforcement training rule (2). It is referred to herein as an integrating pRAM or i-pRAM, and is shown in Figure 4. Thus a real-valued input vector [26] $\underline{x} \in [0,1]^N$ is approximated by the time-average (over some period R) of successive binary input patterns $\underline{i} \in \{0,1\}^N$ (by the real-to-spike-frequency translator [28]:

$$x_j = \frac{1}{R} \sum_{r=1}^{R} i_j(r) \qquad (4)$$

Thus, each of the lines [26] which makes up the vector carries a real value in the range 0 to 1. For each line [26] there is a corresponding address input [5],

and this carries a train of pulses in which the probability of there being, at any given instant, a pulse representing a 1 is equal to the real value on the corresponding line [26]. To put the matter another way, the time average of the pulse train carried by a given line [5] is equal to the value on the corresponding line [26]. The pulse trains on the lines [25] are synchronised with one another. The translator [28] might take various forms, and one possibility is for the translator [28] to be a pRAM itself.

At each time step r = 1...R, $\underline{i}(r)$ selects a particular location in the pRAM [8] using the address inputs [5], resulting in a binary output at [4] denoted herein as $\hat{a}(r)$. These outputs are accumulated in a spike integrator [19] (see Figure 4) whose contents were reset at the start of this cycle. The integrator [19] comprises a counter which counts the number of 1's received over a fixed interval, and, if there is no lookup table [27], for which see below, a device for generating a binary output [21] in dependence on the number counter. This device may itself operate in the manner of a pRAM with a single storage location, i.e. a random number can be added to the contents of the counter and a 0 or 1 generated depending on whether there is an overflow bit. After R time steps the contents of [19] are used to generate the binary i-pRAM

output [21], which is 1 with probability

$$\text{Prob}(a=1 \mid \underline{x}) = \frac{1}{R} \sum_{r=1}^{R} \hat{a}(r) \qquad (5)$$

$$= \sum_{\underline{u}} \alpha_{\underline{u}} \prod_{j=1}^{N} (x_j u_j + \bar{x}_j \bar{u}_j) = \sum_{\underline{u}} \alpha_{\underline{u}} X_{\underline{u}}$$

where $X_{\underline{u}}$ = Prob($\underline{u}$ addressed) is the more general distribution function which replaces the delta function on the right hand side of (1).

As an alternative to averaging over a succession of fixed intervals, each beginning where the last ended, a moving average could be used with the output [21] being generated after the formation of each average.

For some applications it might be desirable to use a function of $\Sigma = \sum_{\underline{u}} \alpha_{\underline{u}} X_{\underline{u}}$ to generate the binary output a:

$$\text{Prob}(a=1 \mid \underline{x}) = f(\Sigma) \qquad (6)$$

f might for example be a sigmoid (with threshold θ and 'inverse temperature' β):

$$f(\Sigma) = \frac{1}{1 + e^{-\beta(\Sigma - \theta)}} \qquad (7)$$

In this case it would be necessary to appropriately transform the contents of the integrator [19] before using the i-pRAM output. This might be achieved

locally in hardware by a lookup table, denoted by [27]. In this case the number of 1's counted by the spike generator [19] is used not to generate a 0 or 1 at the output of the generator [19] itself but as the address of a storage location in the lookup table [27], with each location in the lookup table containing a 0 or 1. Thus the output of the lookup table [27] is a 0 or 1 when addressed by the output of the generator [19].

The i-pRAM just described can be developed further to implement a generalised form of the training rule (2). According to rule (2), the input of a single binary address results in the contents of the single addressed location being modified. However, the i-pRAM can be used to implement a generalised form of the training rule (2) in which the input of a real-valued number causes the contents of a plurality of locations to be modified. This is achieved by using an address counter for counting the number of times each of the storage locations is addressed, thus providing what will be referred to herein as a learning i-pRAM. This generalised training rule is

$$\Delta\alpha_{\underline{u}}(t) = \rho((a - \alpha_{\underline{u}})r + \lambda(\overline{a} - \alpha_{\underline{u}})p\ )(t).X_{\underline{u}}(t) \qquad (8)$$

where $X_{\underline{u}}(t) = \prod_{j=1}^{N} (x_j u_j + \overline{x}_j \overline{u}_j\ )$ replaces the delta function in (2). Thus in the learning i-pRAM case, every location [3] is available to be updated, with the

change proportional to that address's responsibility for the ultimate i-pRAM binary output a(t) (obtained using the algorithm of equation (2).

The $X_{\underline{u}}$'s record the frequency with which addresses have been accessed. A simple modification to the memory section of the pRAM (Figure 1) allows the number of times each address is accessed to be recorded using counters or integrators [22] as shown in Figure 5.

The $X_{\underline{u}}$'s could also be recorded in an auxiliary N-input pRAM, and used to modify the memory contents in a similar manner to Figure 3. However, this method takes 2^N times longer than that using the architecture of Figure 5.

For similar reasons to those considered in connection with Figures 2 and 3, training may be accelerated by letting the learning rate constant, ρ, have an initially high value and tend to zero with time, this being achieved in a similar manner to that described above.

Rule (8) may be further generalised in order to deal with situations in which reward or punishment may arrive an indefinite number of time steps after the critical action which caused the environmental

response. In such delayed reinforcement tasks it is necessary to learn path-action, rather than position-action associations. This can be done by adding eligibility traces to each memory location as shown in Figure 6. These decay exponentially where a location is not accessed, but otherwise are incremented to reflect both access frequency and the resulting i-pRAM action. In this context, "access" means that a storage location with a given address has been accessed, "activity" means that when the storage location was accessed it resulted in the pRAM firing (i.e. a 1 at its output), and "inactivity" means that when the storage location was accessed it did not result in the pRAM firing (i.e. a 0 at its output). The trace e_u in counters or integrators [23] records the number of numbers of occasions on which there was "access and activity" for each given storage location, whilst the trace f_u recorded in counters or integrators [24] records the number of occasions on which there was "access and inactivity" for each given storage location (both are equally important in developing an appropriate response to a changing environment). As in Figure 5, counter or integrator [22] records the total number of times each storage location was accessed. The eligibility traces are initialised to zero at the start of a task, and subsequently updated so that at a time t they have the values

$$e_{\underline{u}}(t) = \delta e_{\underline{u}}(t-1) + \overline{\delta} a(t) X_{\underline{u}}(t) \quad (9a)$$

$$f_{\underline{u}}(t) = \delta f_{\underline{u}}(t-1) + \overline{\delta a}(t) X_{\underline{u}}(t) \quad (9b)$$

where δ is a selected constant, $0 \leq \delta < 1$, and $\overline{\delta} = 1-\delta$. Figure 7 shows the mechanism whereby the eligibility trace $e_{\underline{u}}$ is updated according to equation 9a showing that this feature is hardware-realisable. The current value of $e_{\underline{u}}$ is read from the port [26] and multiplied by the eligibility trace decay rate, δ at [28] using a multiplier [13]. This product is combined using an adder [12] with the product of the pRAM output, a(t) [4], the access count data, $X_{\underline{u}}$ [25] and the complement of the decay rate, δ [29] before written back as $e_{\underline{u}}$ [23] using the write port [27]. This implements equation 9a.

Updating the $f_{\underline{u}}$ term is identical to that above except that it is the inverse of the output, a(t), which is used to implement the equation 9b.

The necessary extension of equation (8), which results in the capacity to learn about temporal features of the environment, is

$$\Delta\alpha_{\underline{u}}(t) = \rho((\overline{\alpha}_{\underline{u}} e_{\underline{u}} - \alpha_{\underline{u}} f_{\underline{u}}) r + \lambda(\overline{\alpha}_{\underline{u}} f_{\underline{u}} - \alpha_{\underline{u}} e_{\underline{u}}) p)(t) \quad (10)$$

When $\delta = 0$, $e_{\underline{u}} = aX_{\underline{u}}$, $f_{\underline{u}} = \overline{a}X_{\underline{u}}$, it may be seen that (10) reduces to the original learning i-pRAM training

rule (8).

In addition to updating the eligibility traces (shown in Figure 7) the memory contents, $\alpha_{\underline{u}}$ are modified so that learning behaviour may be implemented. Figure 8 shows the operations required in addition to those of Figure 7 in order to implement equation 10. Multiplier [31] forms the product of $e_{\underline{u}}$ and $\bar{\alpha}_{\underline{u}}$ and multiplier [32] forms the product of $f_{\underline{u}}$ and $\bar{\alpha}_{\underline{u}}$. Multiplier [33] forms the product of $e_{\underline{u}}$ and $\alpha_{\underline{u}}$ and multiplier [34] forms the product of $f_{\underline{u}}$ and $\alpha_{\underline{u}}$. The product formed by multiplier [33] is subtracted from the product formed by multiplier [32] in the subtractor [35]. The product formed by multiplier [34] is subtracted from the product formed by multiplier [31] in the subtractor [36]. The output of the subtractor [35] is multiplied by a penalty factor p which is an input from the environment to the multiplier [37] at [39]. The output of the subtractor [36] is multiplied by a reward factor r which is an input from the environment to the multiplier [38] at [40]. The outputs of the multipliers [37] and [38] are added to the original memory contents at [19] using the adder [12]. The output from the adder [12] is written back into the memory using the write port [10] and the memory is thereby updated. The operations described implement the training rule described in equation 10.

An alternative to the training rule of equation (8) is a rule which may take account more realistically of the behaviour of the whole i-pRAM. This alternative is expressed by

$$\Delta\alpha_{\underline{u}}(i) = \rho\{[\bar{\alpha}_{\underline{u}}(i)\bar{g})a_i - (\alpha_{\underline{u}}(i)g)\bar{a}_i]r + \lambda[(\bar{\alpha}_{\underline{u}}(i)\bar{g})\bar{a}_i - (\alpha_{\underline{u}}(i)g)a_i]p\}X_{\underline{u}}$$

where g is a suitable function of $\Sigma_{\underline{u}}\alpha_{\underline{u}} X_{\underline{u}}$ such as, for example

$$g(x) = \frac{1}{1 + e^{-\beta(x - \theta)}}$$

Where eligibility traces are added, this becomes

$$\Delta\alpha_{\underline{u}}(i) = \rho\{[(\bar{\alpha}_{\underline{u}}(i)g)e_{\underline{u}} - (\alpha_{\underline{u}}(i)g)f_{\underline{u}}]r + \lambda[(\bar{\alpha}_{\underline{u}}(i)g)f_{\underline{u}} - (\alpha_{\underline{u}}(i)g)e_{\underline{u}}]p\}$$

In the various aspects of the invention described herein, the devices are described as being realised in dedicated hardware. It will be appreciated that the invention can alternatively be realised in software, using a conventional digital computer to simulate the hardware described, and the present application is intended to encompass that possibility. However, software simulation is unlikely to be practical except for very small networks and the hardware approach is

much more practical for larger and therefore more interesting networks.

Also is should be noted that other hardware realisations are possible, for example using VLSI technology.

CLAIMS

1. A device for use in a neural processing network comprising a memory having a plurality of storage locations at each of which a number representing a probability is stored; means for selectively addressing each of the storage locations to cause the contents of the location to be read to a comparator; a noise generator for inputting to the comparator a random number representing noise; and means for causing to appear at the output of the comparator an output signal having a first or second value depending on the values of the numbers received from the addressed storage location and the noise generator, the probability of the output signal having a given one of the first and second values being determined by the number at the addressed location; means for receiving from the environment signals representing success or failure of the network; means for changing the value of the number stored at the addressed location if a success signal is received in such a way as to increase the probability of the successful action; and means for changing the value of the number stored at the addressed location if a failure signal is received in such a way as to decrease the probability of the unsuccessful action.

2. A device according to claim 1, wherein the random numbers and the numbers at the storage location have the same number of bits, and wherein the comparator is operable to add the values of the random number received and the number received from the addressed location, the output signal having the said first or second value depending on whether or not the addition results in an overflow bit.

3. A device according to claim 1 or 2, wherein the means for increasing or decreasing the value of the number stored at the addressed location operates according to a training rule described by the equation

$$\Delta\alpha_{\underline{u}}(t) = \rho((a - \alpha_{\underline{u}})r + \lambda(\bar{a} - \alpha_{\underline{u}})p)(t).\delta(\underline{u} - \underline{i}(t))$$

where r(t) and p(t) are success and failure signals $\in$ (0,1) respectively received from the environment at time t,

a(t) is the value of the output signal of the comparator $\in$ (0,1),

ρ and λ are constants $\in$ [0,1],

$\alpha_{\underline{u}}$ is the probability represented by the number stored at the address location $\underline{u}$, and

$\delta(\underline{u}-\underline{i}(t))$ is a delta function expressing the fact that only the number stored at the addressed location is modified.

4. A device according to claim 3, modified in that the value of ρ is decremented at each successive training operation by an amount which progressively decreases.

5. A device according to any preceding claim, further comprising a real-number to digital converter which receives a plurality of real-valued numbers each in the range of 0 to 1 and produces at its output a corresponding plurality of synchronised parallel pulse trains which are applied to the respective address lines of the memory to define a succession of storage location addresses, the probability of a pulse representing a 1 being present in an address on a given address line being equal to the value of the real-valued number from which the pulse train applied to that address line was derived; an address counter for counting the number of times each of the storage locations is addressed; means for increasing or decreasing the values of the numbers stored at the addressed locations in dependence on the number of times each of the storage locations is addressed, as counted by the address counter; and an integrator for integrating the output signals from the comparator.

6. A device according to claim 5, further

comprising an output generator connected to the integrator and having an output at which a given one of two values appears as a function of the integrated value produced by the integrator.

7. A device according to claim 6, wherein the output generator contains a look-up table for generating the given one of the two values as a function of the integrated value produced by the integrator.

8. A device according to any one of claims 5 to 7, wherein the increase or decrease, $\Delta\alpha_{\underline{u}}(t)$, in the value of the number stored at a given location is given by the equation:

$$\Delta\alpha_{\underline{u}}(t) = \rho\left((a - \alpha_{\underline{u}})r + \lambda(\bar{a} - \lambda(\bar{a} - \alpha_{\underline{u}})p \right)(t).X_{\underline{u}}(t)$$

where $X_{\underline{u}}(t) = \prod_{j}^{N} (x_j u_j + \bar{x}_j \bar{u}_j)$.

9. A device according to any one of claims 5 to 7, with the addition of two further counters for each address location, the contents of one of said further counters being subject to an increment on each occasion when the storage location concerned is accessed and the output signal of the device has the said first value, and the contents of the other of said further counters being subject to an increment on each occasion when the

storage location concerned is accessed and the output signal of the device has the said second value, the contents of both said further counters being subject to a decay factor each time any storage location is accessed, the values of the numbers stored at the addressed locations being increased or decreased in dependence on the contents of the said further counters.

10. A device according to claim 9, wherein the contents $e_{\underline{u}}(t)$ and $f_{\underline{u}}(t)$ of the said further counters at time t are given by

$$e_{\underline{u}}(t) = \delta e_{\underline{u}}(t-1) + \overline{\delta} a(t) X_{\underline{u}}(t)$$

$$f_{\underline{u}}(t) = \delta f_{\underline{u}}(t-1) + \overline{\delta a}(t) X_{\underline{u}}(t)$$

where δ is a selected constant, $0 \leq \delta < 1$, and $\overline{\delta} = 1-\delta$.

11. A device according to claim 10, wherein the increase or decrease $\Delta\alpha_{\underline{u}}(t)$, in the value of the number stored at a given location is given by the equation:

$$\Delta\alpha_{\underline{u}}(t) = \rho\left((\overline{\alpha_{\underline{u}} e_{\underline{u}}} - \alpha_{\underline{u}} f_{\underline{u}}) r + \lambda (\overline{\alpha_{\underline{u}} f_{\underline{u}}} - \alpha_{\underline{u}} e_{\underline{u}}) p \right)(t)$$

12. A device according to any preceding claim, wherein the memory is a random access memory.

13. A device for use in a neural processing network comprising a memory having a plurality of

storage locations at each of which a number representing a probability is stored; means for selectively addressing each of the storage locations to cause to appear at the output of the device an output signal having a first or second value depending on the value of the number at the addressed storage location, the probability of the output signal having a given one of the first and second values being determined by the number at the addressed location; means for receiving from the environment signals representing success or failure of the network; means for changing the value of the number stored at the addressed location if a success signal is received in such a way as to increase the probability of the successful action; and means for changing the value of the number stored at the addressed location if a failure signal is received in such a way as to decrease the probability of the unsuccessful action.

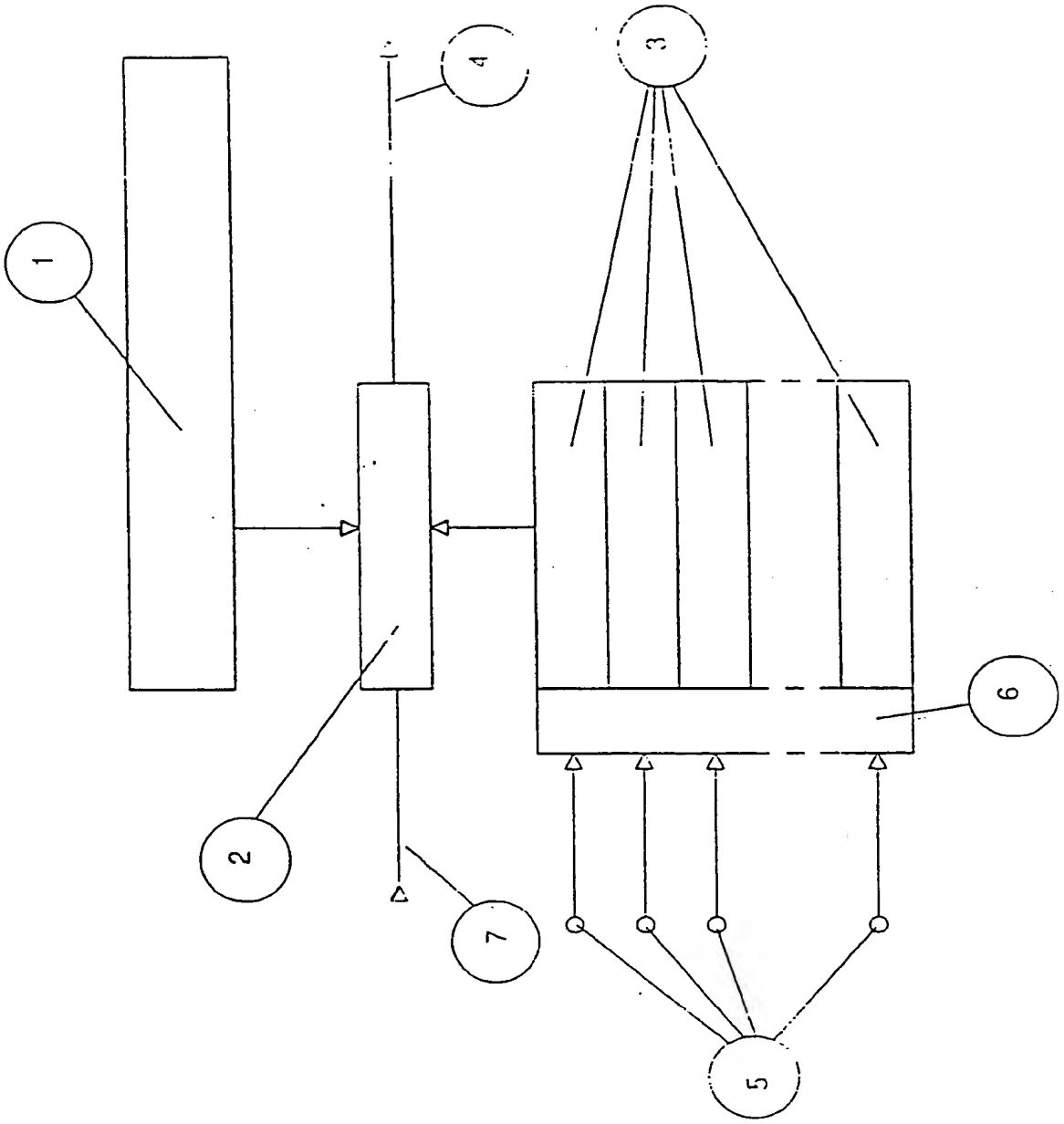


Figure 1

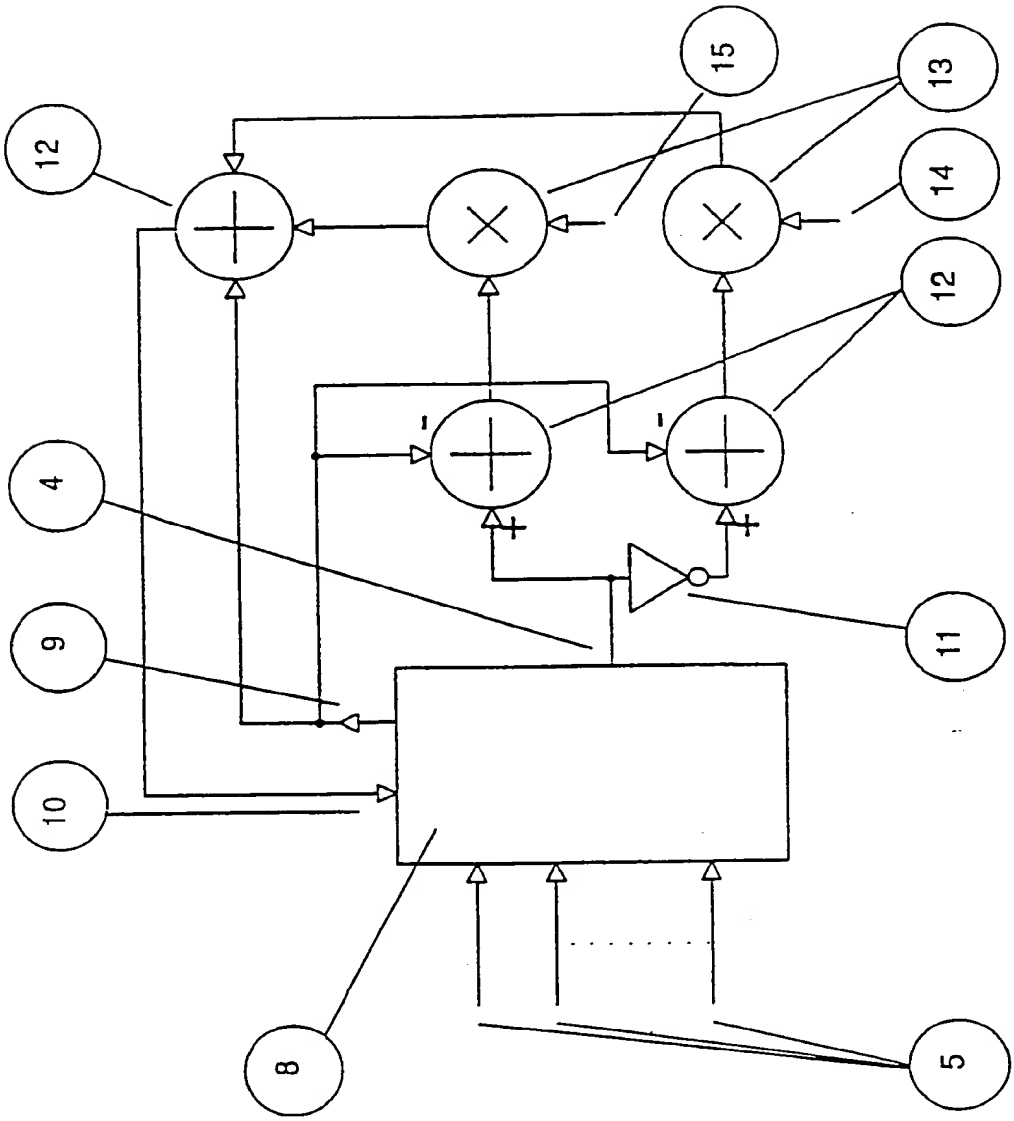


Figure 2

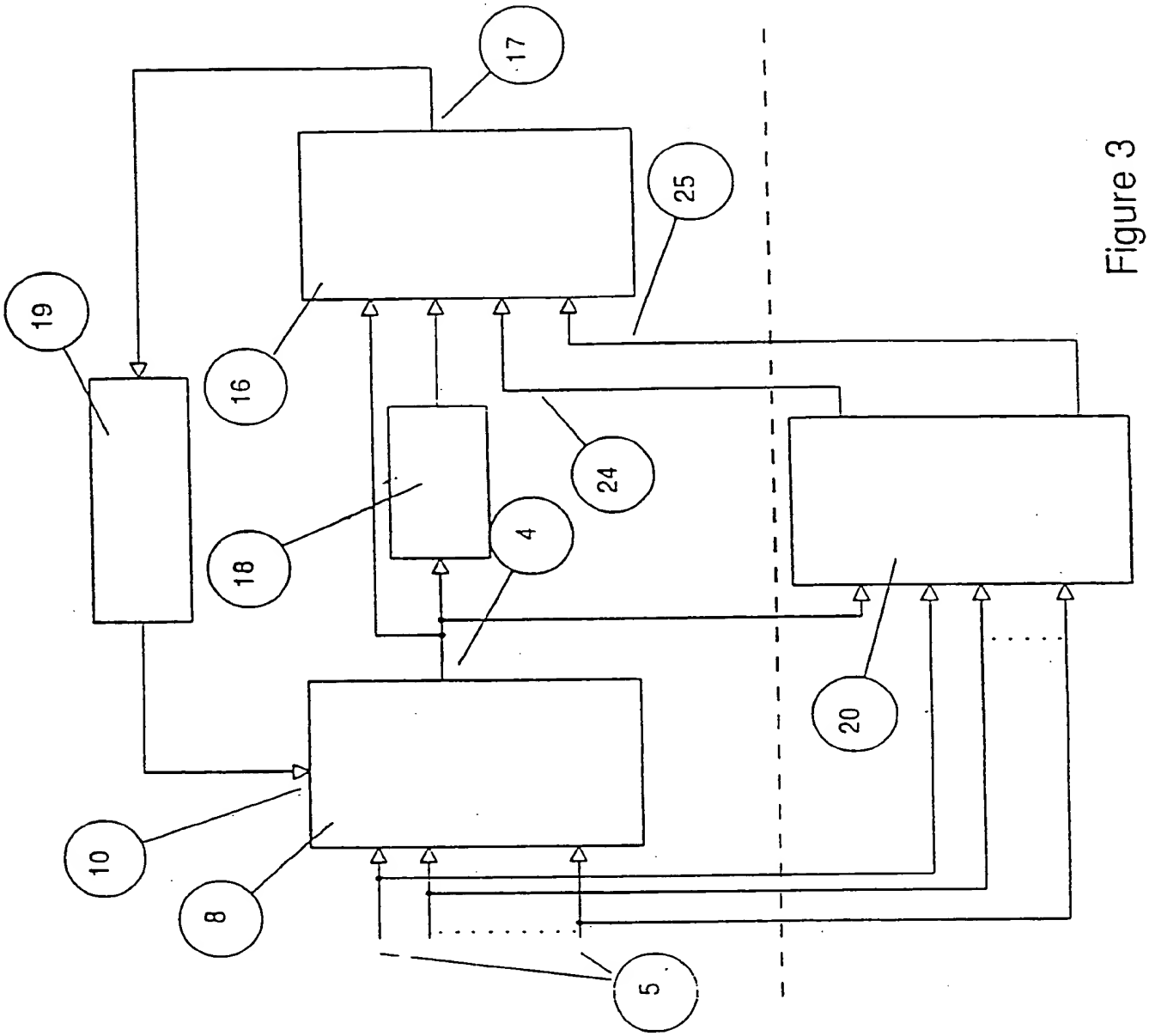


Figure 3

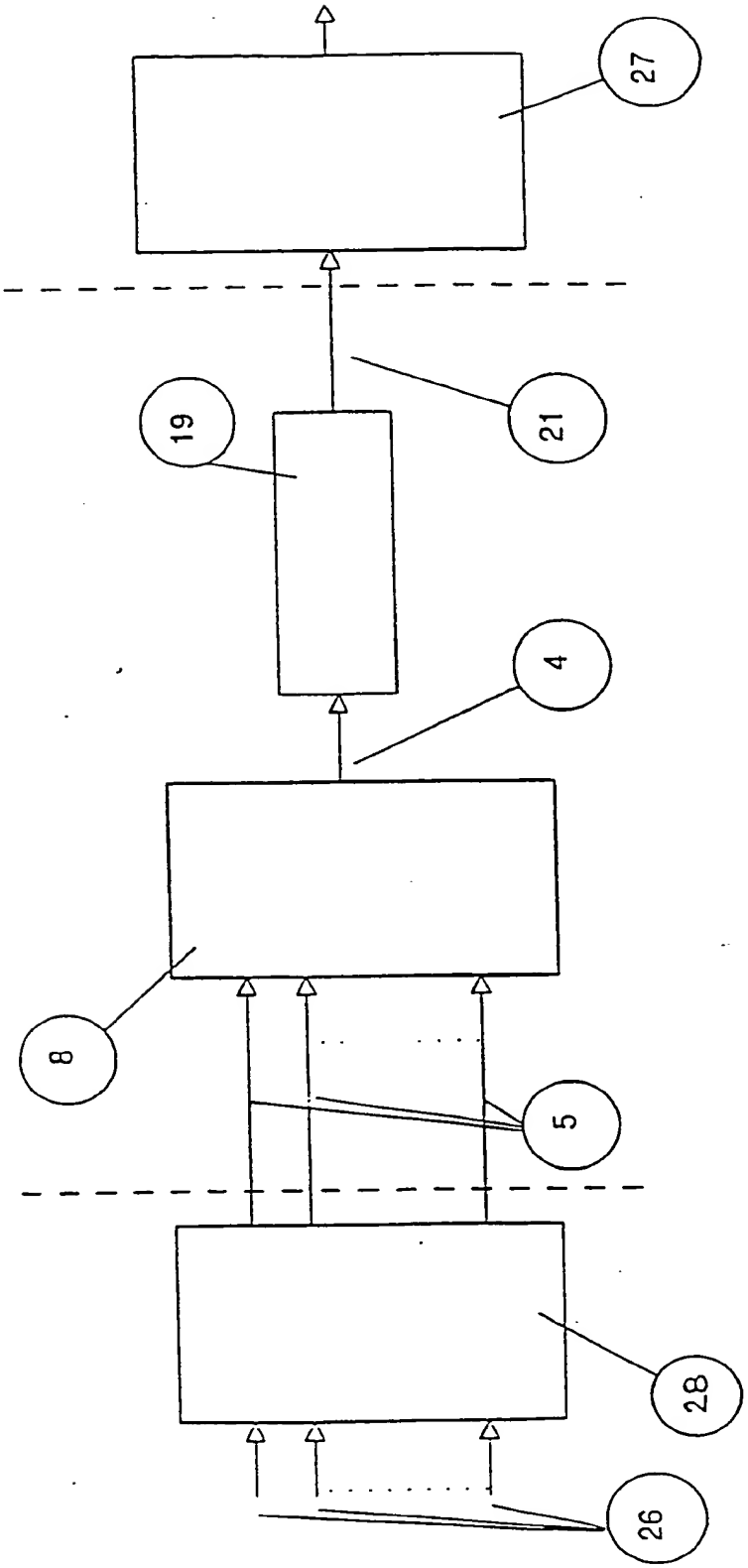


Figure 4

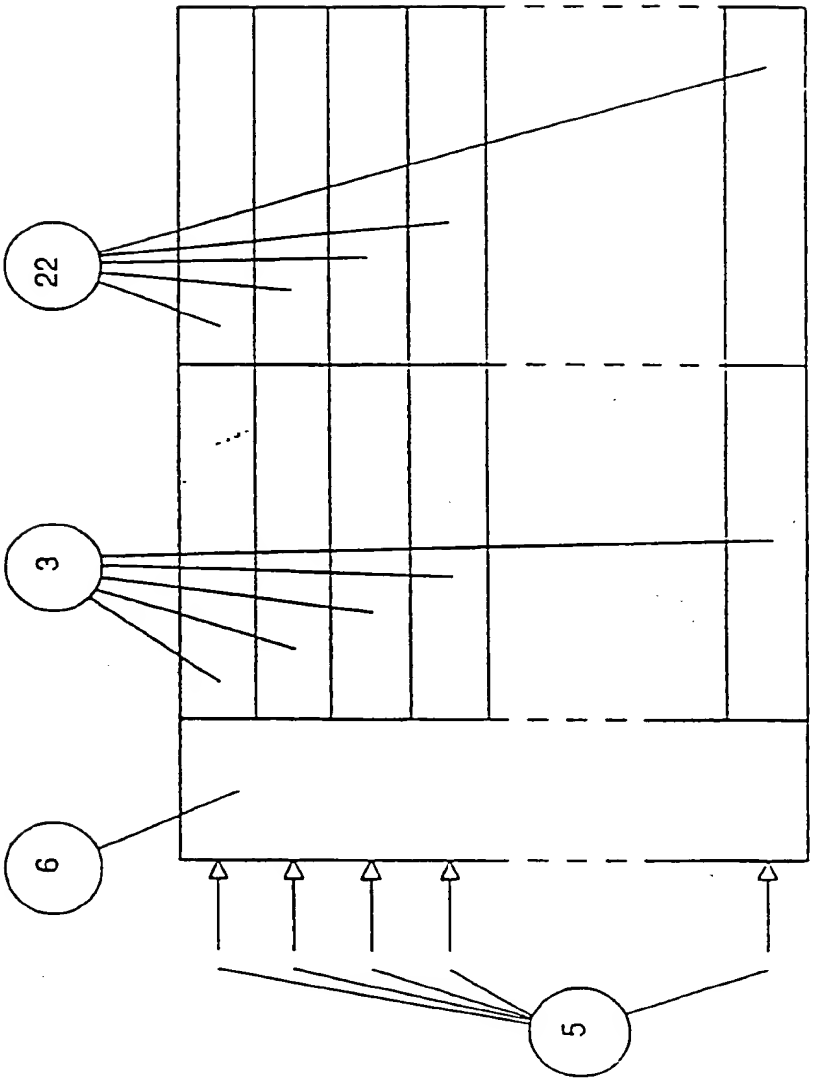


Figure 5

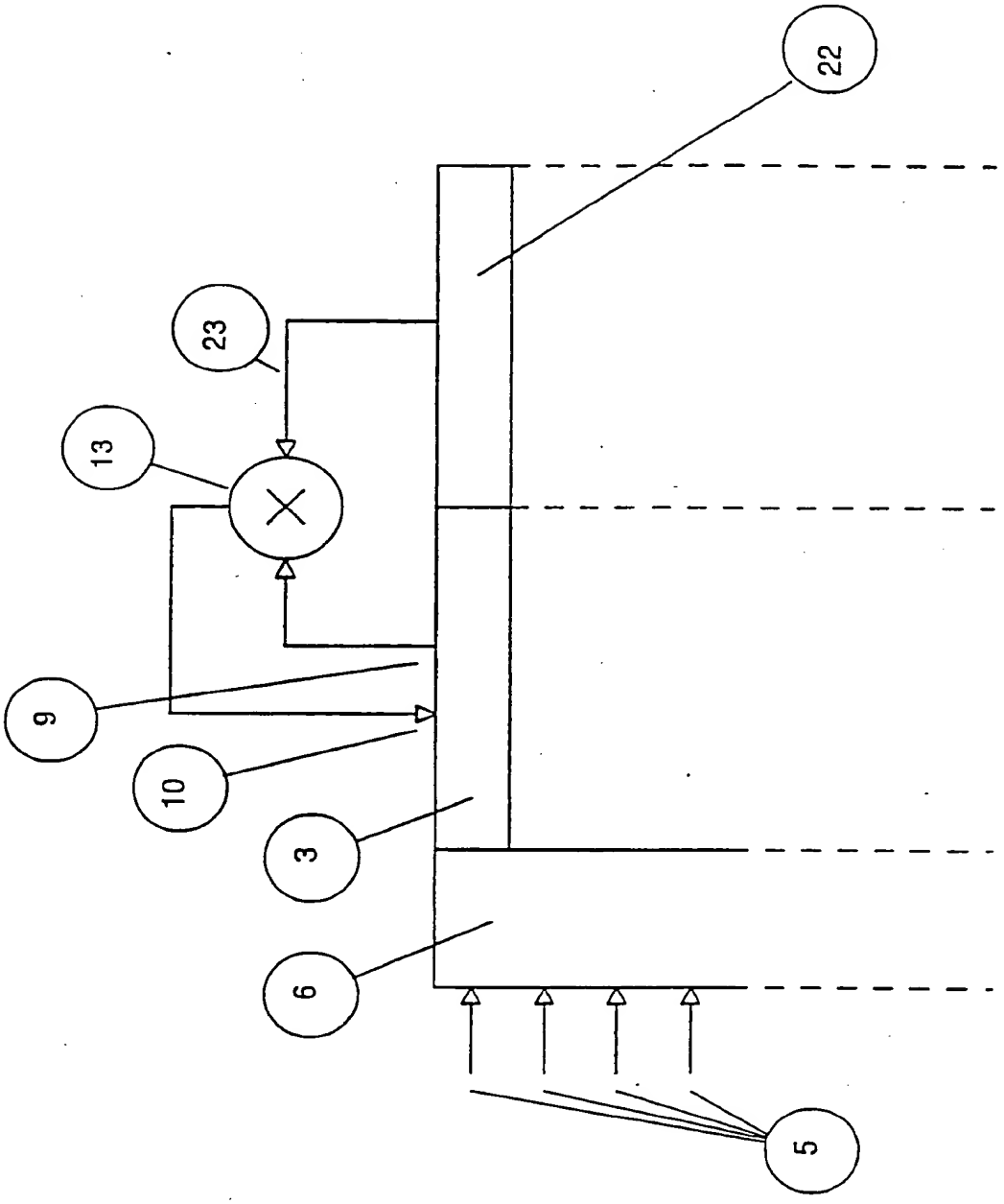


Figure 6

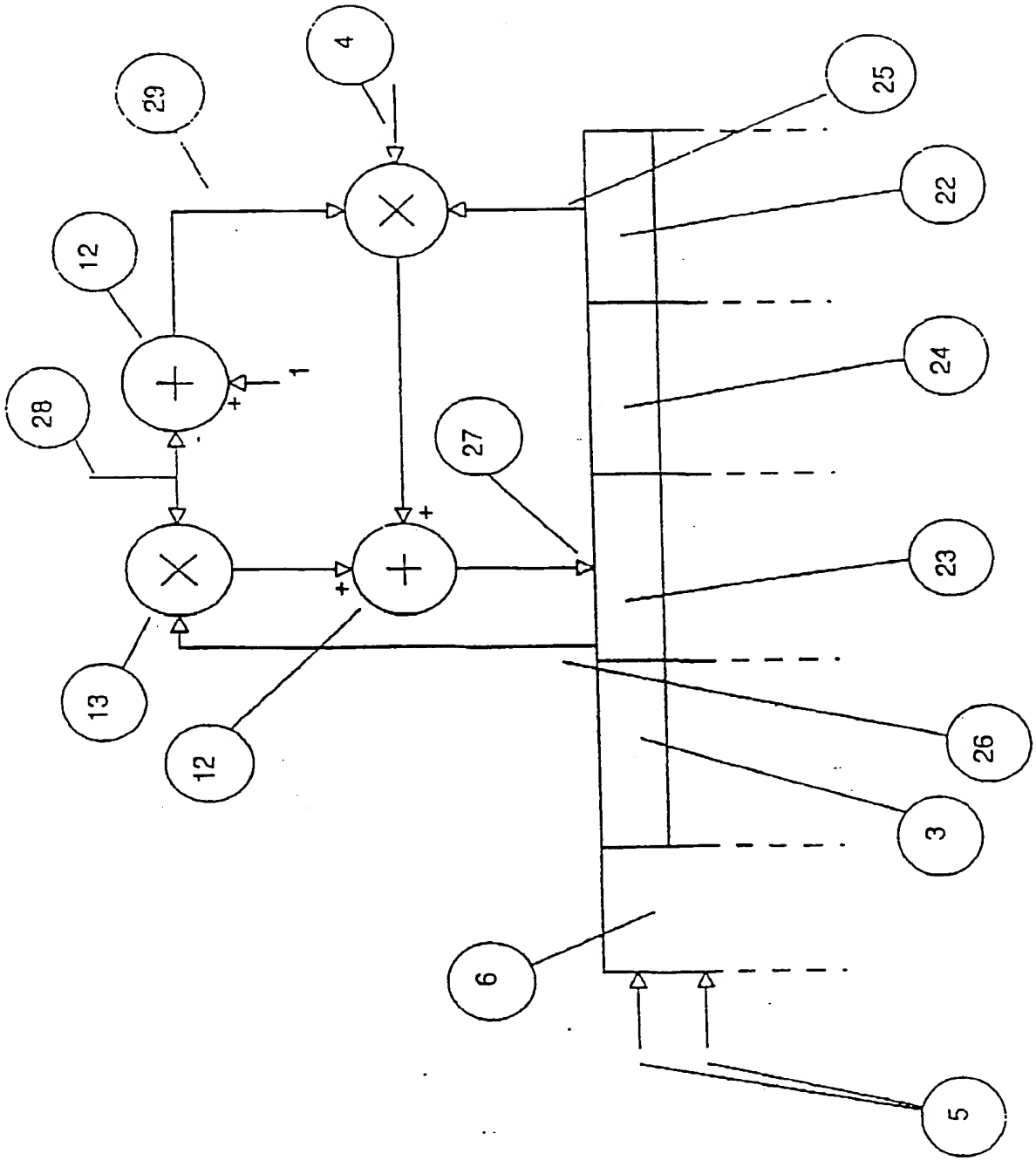


Figure 7

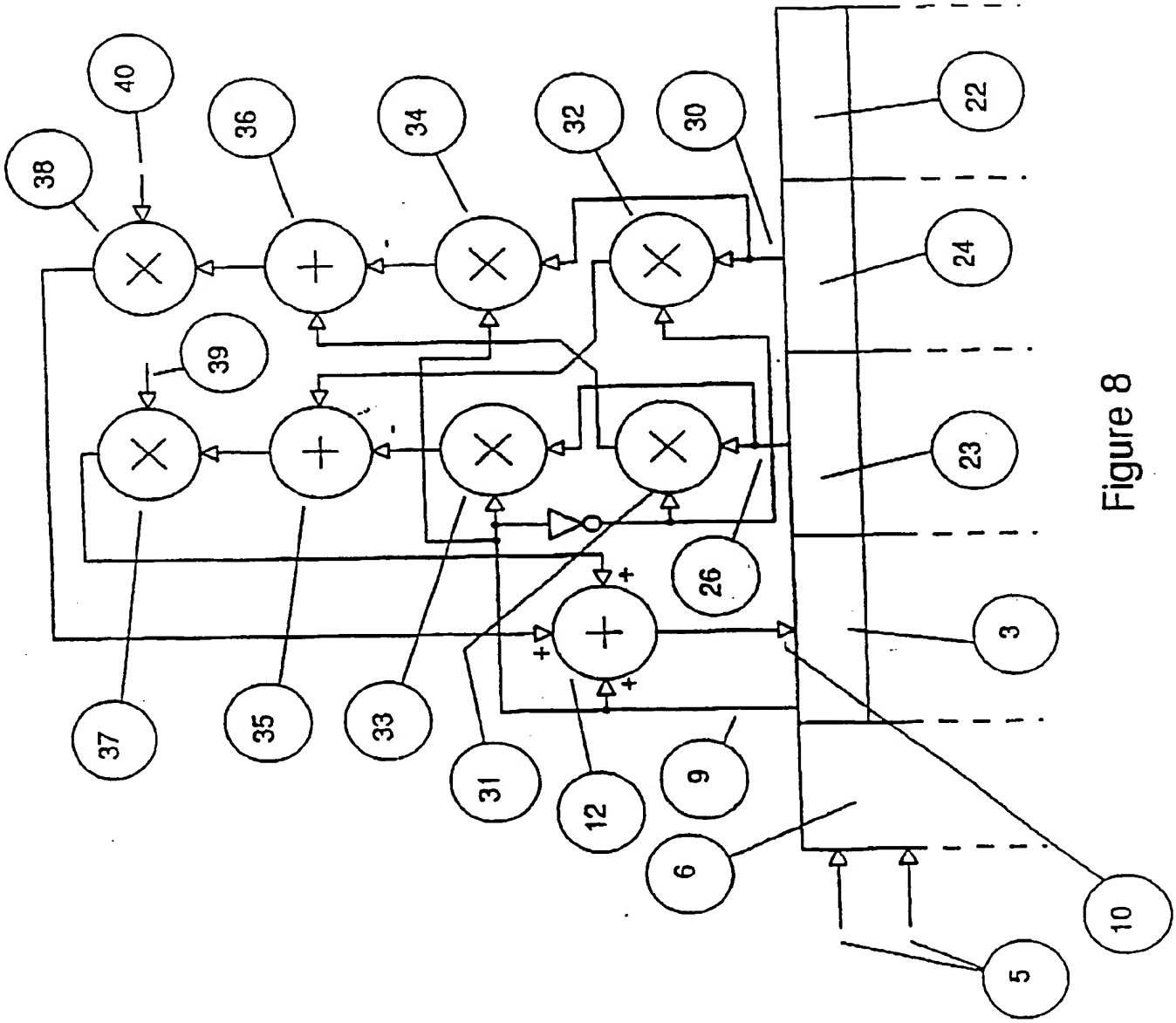


Figure 8

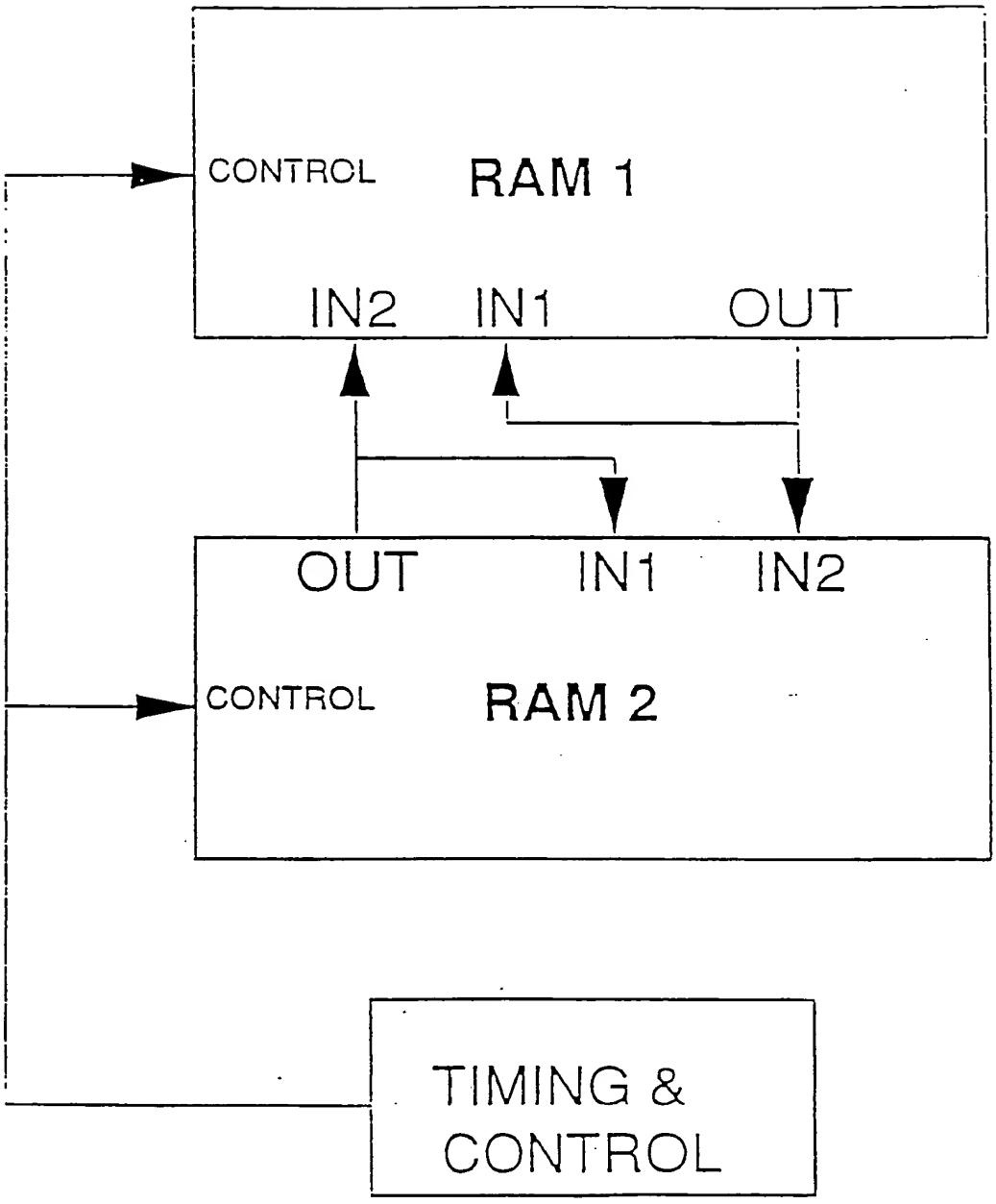

CONTROL
RAM 1
IN2
IN1
OUT
OUT
IN1
IN2
CONTROL
RAM 2
TIMING &
CONTROL

Figure 9

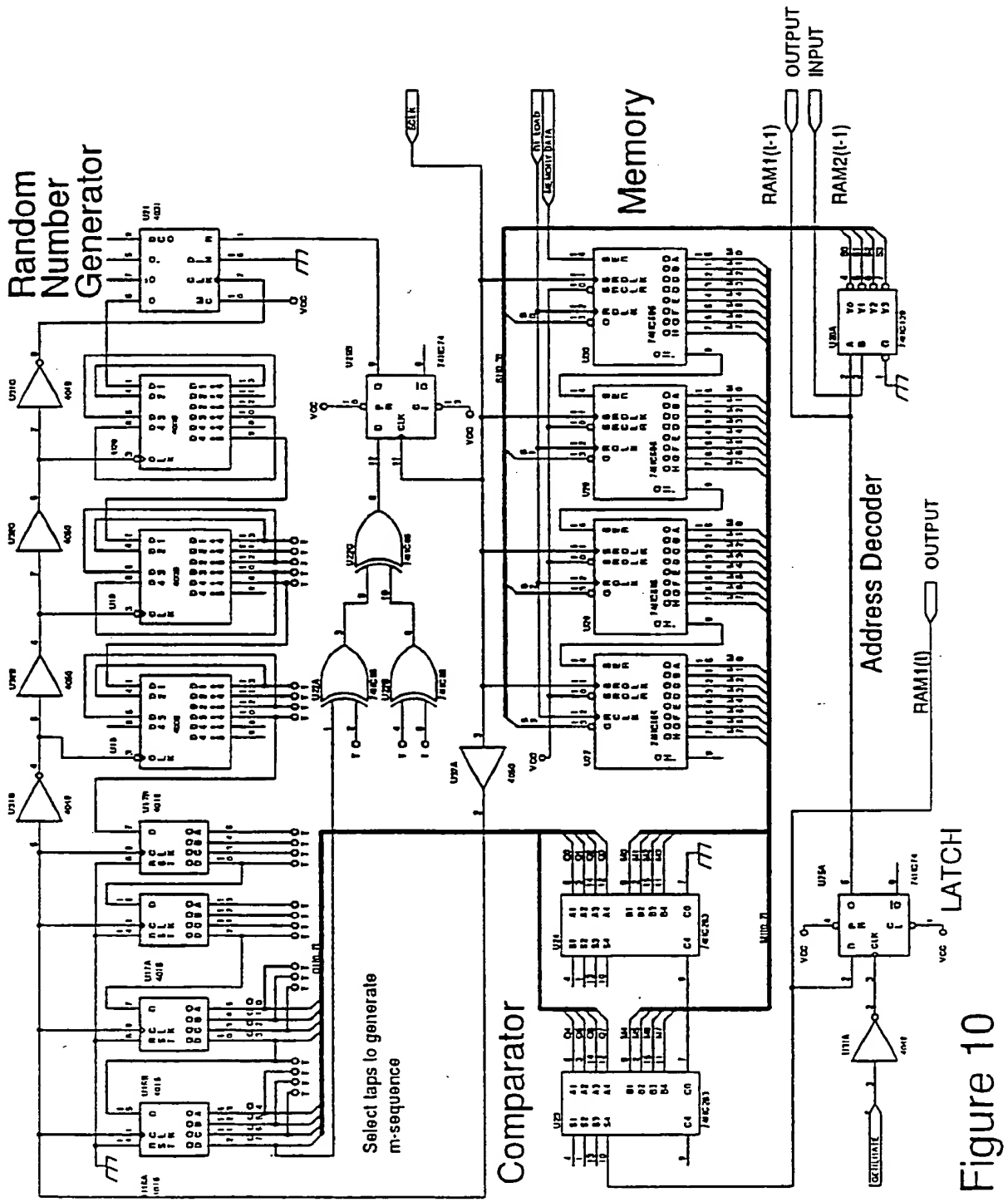

Random Number Generator
Memory
OUTPUT
INPUT
RAM1(I-1)
RAM2(I-1)
Address Decoder
RAM1(I)
OUTPUT
LATCH
Comparator
Select taps to generate m-sequence

Figure 10

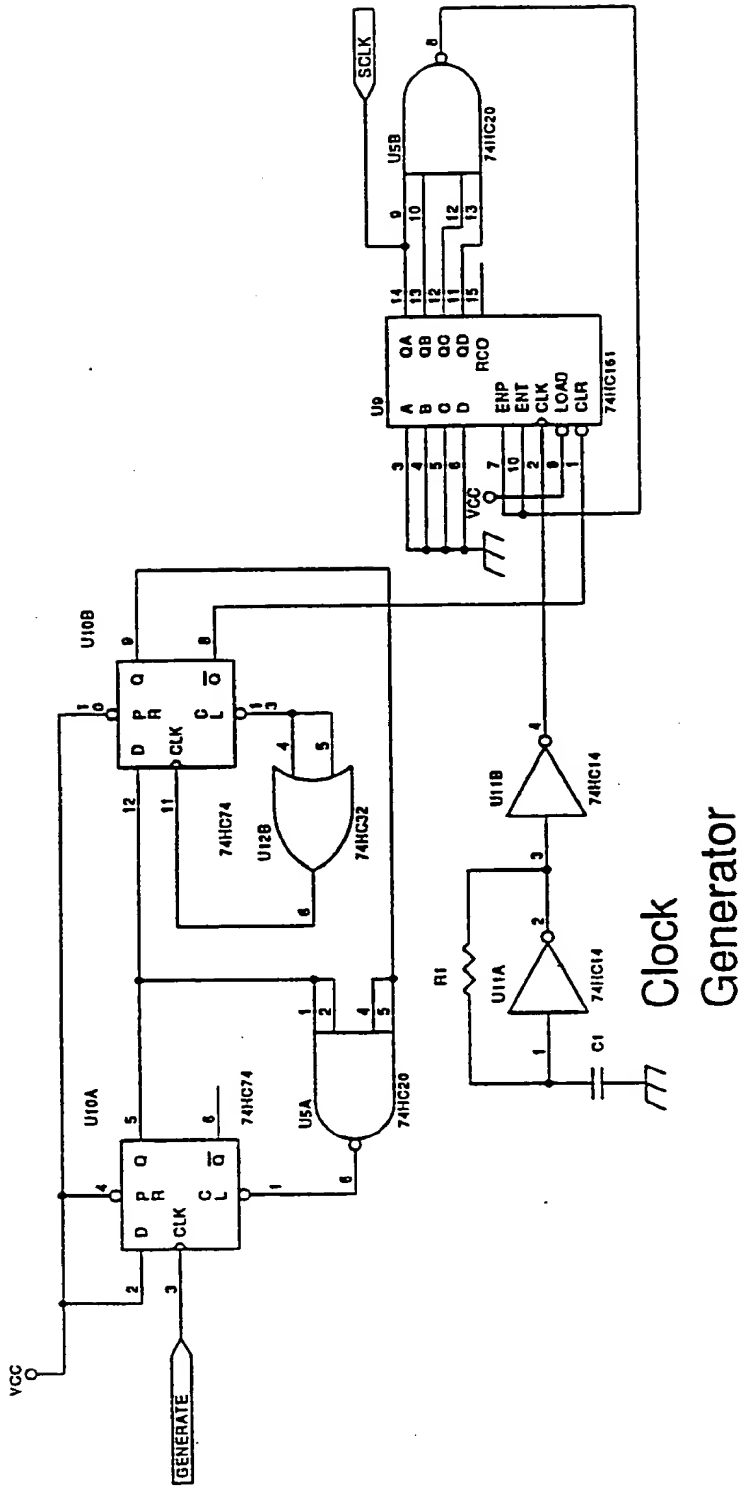

SCLK
U5B
74HC20
U9
74HC161
QA
QB
QC
QD
RCO
A
B
C
D
ENP
ENT
CLK
LOAD
CLR
VCC
U10B
74HC74
U12B
74HC32
U11B
74HC14
R1
U11A
74HC14
C1
U10A
74HC74
U5A
74HC20
VCC
GENERATE
Clock
Generator

Figure 11